8086 Microprocessor Laboratory Experiments

Experiment 4: The Stack

Murtadha Hssayeni, Ph.D.

m.hssayeni@uobabylon.edu.iq

The Stack

- The **stack** is an area of memory for **keeping temporary addresses and data.**
- For 8086, the stack is 64KByte long, and it is organized as 32K words from the software point of view.
  - The stack pointer (SP) and base pointer (BP) are used as offset beside the stack segment register (SS).
- The stack is used by CALL instruction to **keep return address for procedure**
  - RET instruction gets this value from the stack and returns it (offset) to IP register.
  - When INT instruction calls an interrupt, it stores code segment and offset in the stack.
  - IRET instruction is used to return from interrupt call.
- We can also use the stack to **keep any other data,**
- There are two instructions that work with the stack:
  - **PUSH** - stores 16 bit value in the stack.
  - **POP** - gets 16 bit value from the stack.

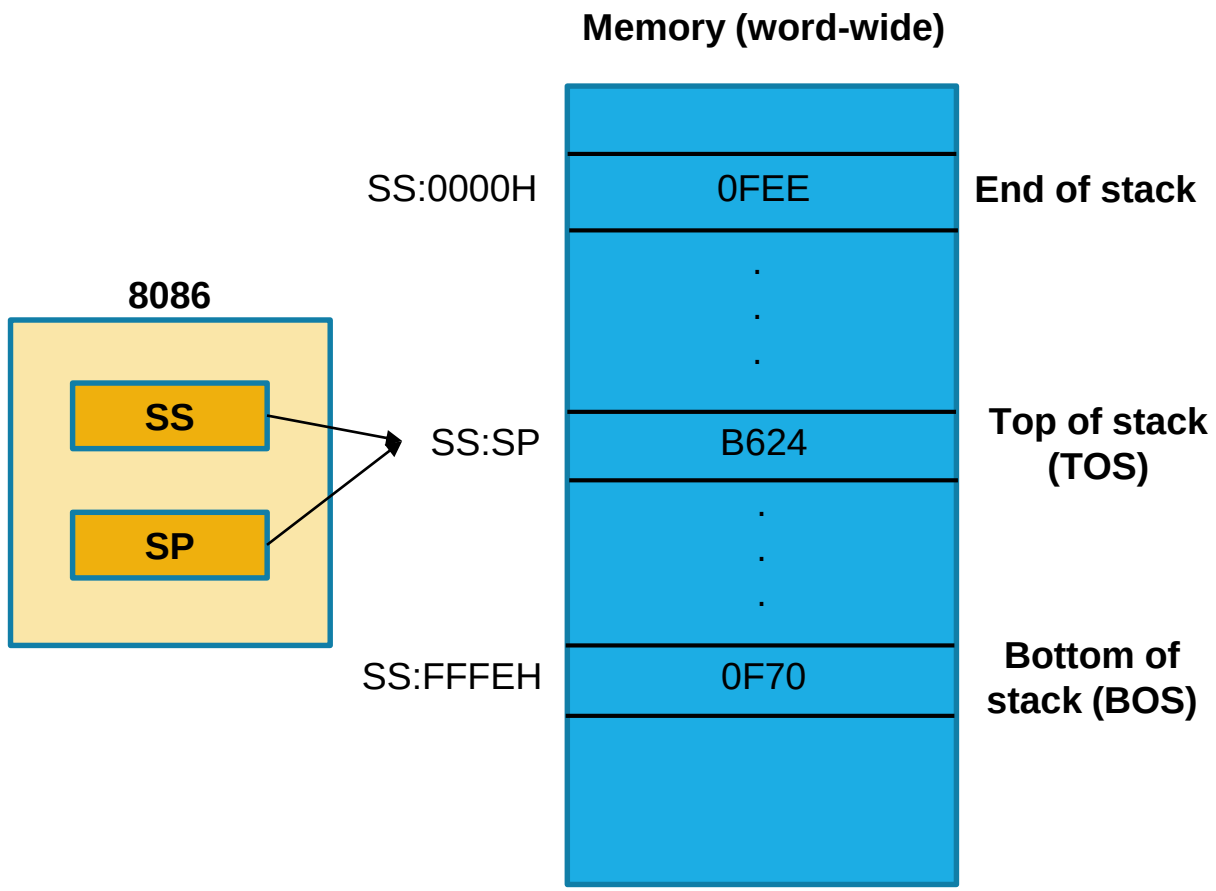
The Stack: Instructions Syntax

- Syntax for **PUSH** instruction:
  - *PUSH REG*
  - *PUSH SREG*
  - *PUSH memory*
  - *PUSH immediate*
  - Pushing a word-wide data.
- REG: AX, BX, CX, DX, DI, SI, BP, SP.
- SREG: DS, ES, SS, CS.
- Memory: [BX], [BX+SI+7], 16 bit variable, etc...
- Immediate: 5, -24, 3Fh, 10001101b, etc...

- Syntax for **POP** instruction:
  - *POP REG*
  - *POP SREG*
  - *POP memory*
- REG: AX, BX, CX, DX, DI, SI, BP, SP.
- SREG: DS, ES, SS, (except CS).
- Memory: [BX], [BX+SI+7], 16 bit variable, etc...

The Stack: LIFO

- The stack uses **Last-In First-Out (LIFO) algorithm**
  - **T**his means that the first pushed value would be the last popped value from stack memory.
- It is very important to do equal number of PUSHs and POPs
- PUSH and POP instruction are especially useful because we don't have too many registers to operate with:
  - Store original value of the register in stack (using PUSH).
  - Use the register for any purpose.
  - Restore the original value of the register from stack (using POP).

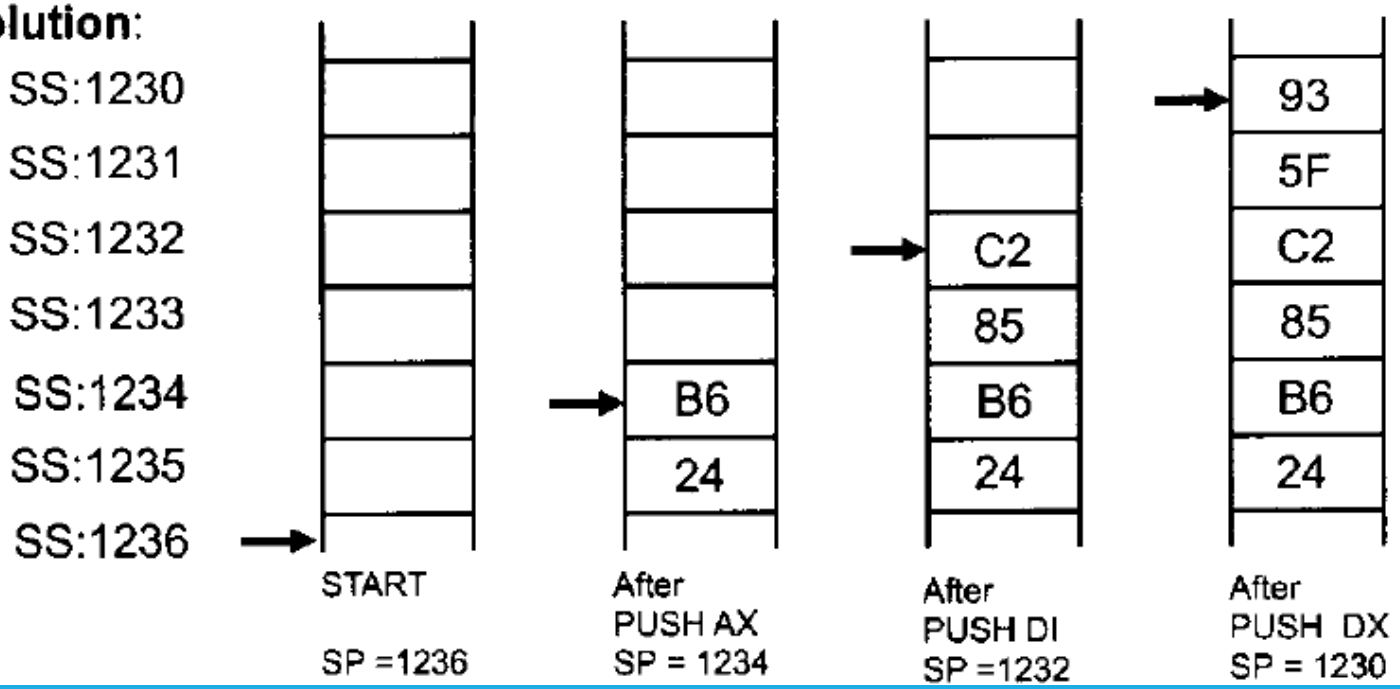


The Stack: Example 1

Assuming that SP = 1236, AX = 24B6, DI = 85C2, and DX = 5F93, show the contents of the stack as each of the following instructions is executed:

```
PUSH  AX
PUSH  DI
PUSH  DX
```

Solution:

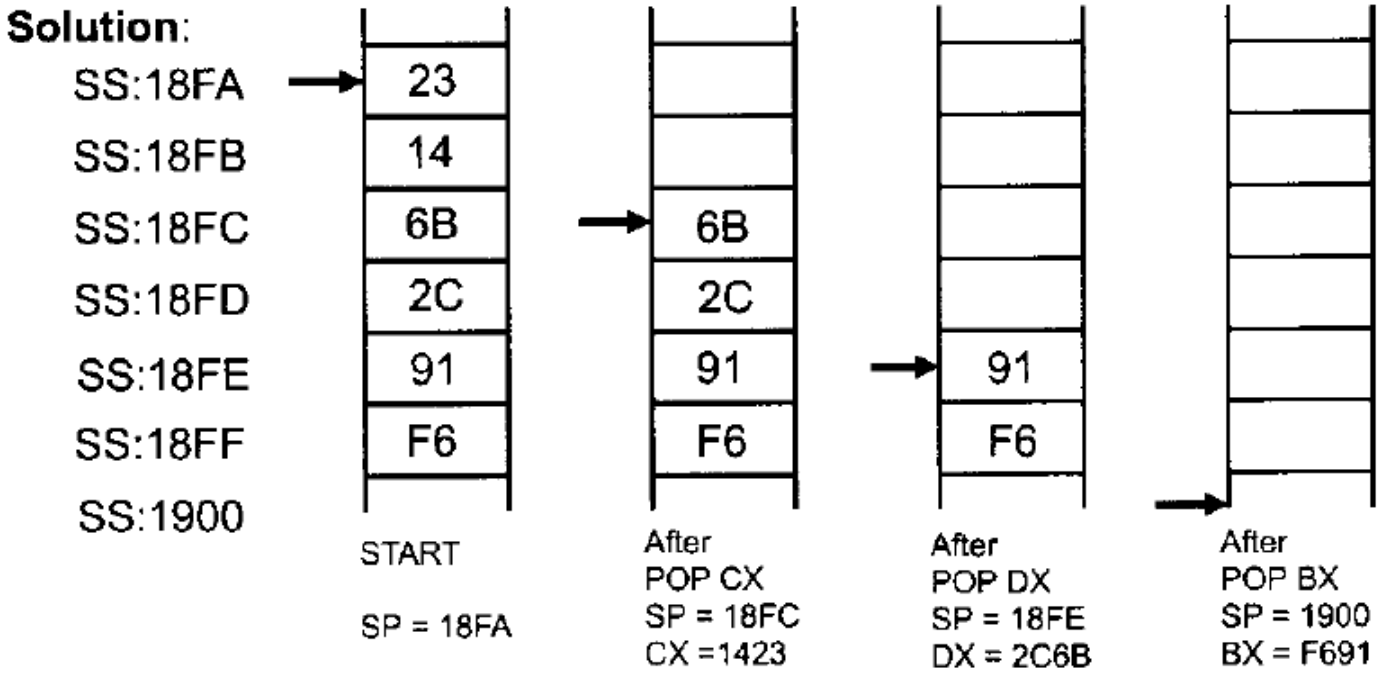


The Stack: Example 2

Assuming that the stack is as shown below, and SP = 18FA, show the contents of the stack and registers as each of the following instructions is executed:

```
POP   CX
POP   DX
POP   BX
```

Solution:



Example 1

Write a 8086 program to do the following:

- Store the value 1234H to the register AX.
- Store the value 5678H to the register BX.
- Store the register AX to the stack.
- Copy the value of the register BX to AX.
- What is the logical address of TOS?
- Restore the value of the register AX from the stack.

```
org 100h
MOV AX, 1234H
MOV BX, 5678H
PUSH AX
MOV AX, BX
; do some work on AX then retrieve old value.
; Logical address of TOS is SS:SP 0700:FFFC
POP AX
ret
```

Procedure

1. Write a simple program to push the following values to the stack: AA00H, BB11H, and 33FFH. Then Pull 33FFH to DX.
2. What is the logical address of the TOS and BOS?
3. What is the physical address of TOS?
4. Pull AA00H to CX

Discussion

1. Write a program to Store the value 1234H to the register AX. Store the value 5678H to the register BX. Next swap the values of AX and BX using the Stack.
2. Run the following code:

```
ORG 100H

PUSH 0FFEEH
CALL add1
POP BX
; other operations

add1:
ADD AX, CX
RET

RET
```

3. What do you notice about the stack and the SP when executing
    1. The CALL instruction.
    2. The RET instruction.